AUTOMATIC CHORD RECOGNITION
CSC 475 / SENG 480B - Music Information Theory- Fall 2020

Chris Clarke
MUCS
V00845052
cdclarke@uvic.ca

Grant Hames
SENG
V00857826
hamesg@uvic.ca

Matthew Haussmann
MUCS
V00827158
haussmannm@gmail.com

William Sease
CSC/MUS
V00857661
william.sease@shaw.ca

ABSTRACT

Development into the algorithms used for Automatic Chord Recognition has proceeded collectively for the last 20 years. In our work, we review the MIR techniques that have been used in recognition algorithms over the years and implement these components with additional novel changes to explore the workings of ACR systems.

1. INTRODUCTION

Chords and chord sequences are fundamental components of music that contribute to the structure and progression of a song. The recognition of chords in music is a skill that musicians develop through a combination of musical theory training—to understand the rules and best practices in music, and ear training—to recognize these concepts while listening to music. The ability to identify and categorize chords accurately has many data mining applications such as the classification of music, tracking similarities between pieces of music, or creating songs that feel familiar to listeners. The identification of chords is difficult and potentially subjective as experts cannot always agree on the chords in a piece.

Automatic Chord Recognition (ACR) is the process of using Music Information Retrieval (MIR) techniques to detect and identify the chords within music. The field of MIR has been developing methods of chord recognition for at least 20 years [1]. The need for ACR comes in part from the difficulty of identifying chords by hand, and that the process can be ambiguous and subjective by nature. Successful ACR algorithms are highly desired in the aforementioned data mining applications to perform difficult recognition quickly at high scale for large music libraries. Furthermore, other practical applications for ACR could be the automatic generation of lead sheet notation for musicians that do not have the ability or the time to transcribe by ear.

2. RELATED WORK

In November 2019 a paper titled *"20 Years of Automatic Chord Recognition From Audio"* [1] reviewed the history of ACR in the field of MIR, outlining multiple techniques that were developed and used together to improve the performance and accuracy of ACR over the years. This paper also describes a more "data-driven" approach that uses machine learning and large amounts of data with little use of MIR techniques. While data-driven models are growing in popularity, there is much to learn from the history of rule-based ACR systems, which combined multiple MIR components into one cohesive system.

Inspired by this paper, we have designed and realized the implementation of a system based-on MIR principles. This system extends MIR components outlined in [1] by following a modular approach that incorporates ideas from previous MIR techniques shown to be successful in ACR. Our system follows this design instead of the neural network-based approach. In related works, chord progression analysis and prediction are often part of an ACR system, even one without neural-networks. These use either probability-based or knowledge-based algorithms with Hidden Markov Models [9] to smooth the sequences of chord labels for a piece of music. However, this has not been realized by our system due to time constraints and difficulties faced by one of the members in the team.

Modular ACR approaches have been reviewed before in related works such as the Zenz & Rauber [8] implementation. However, our system introduces a novel chroma feature that specifically extracts and makes use of the bass note of the chord to improve recognition accuracy. In the original pitch class profile proposed by Fujishima [10], a twelve dimension vector representing the intensities of the twelve semitone pitch classes is used. This has been the standard system in ACR and is seen in all implementations reviewed in [1], but does not make use of the bass note, despite being an essential feature of a chord.

3. BACKGROUND

Musical chords are a harmonic structure consisting of a stack of individual pitches. Chords are hierarchical by nature of their stacked intervals, meaning one chord could contain the all notes of a different but related chord. Intervals are built of a number of semitones (half-steps) and are usually stacked by major or minor thirds, which consist of four and three semitones respectively. When a chord has only three notes in it it is called a triad. When there are four it is referred to as a seventh chord.

Our system has implemented only searches for common triads and sevenths. Common triads are major, minor, diminished, and augmented chords.

- Major triads consist of a minor third on top of a major third.
- Minor triads are a major third on top of a minor third.
- Diminished triads are two minor thirds.
- Augmented triads are two major thirds.

We look for three types of seventh chords in our system: dominant sevenths, major sevenths, and minor sevenths.

  - A dominant seventh is a major triad with an additional minor third on top.
  - A major seventh is a major triad with a major third on top.
  - A minor seventh is a minor triad with a minor third on top.

This is shown in Figure 1 below:

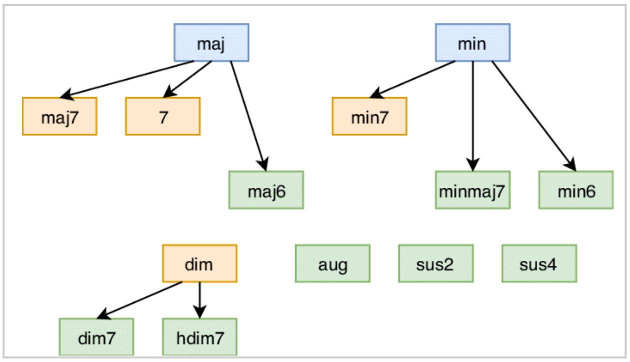


Figure 1: Hierarchies of Recognizable Chords

A musical octave has twelve semitones forming what is known as the chromatic scale, repeating after the twelfth note. This is derived from the fact that an octave is double the fundamental frequency of the first pitch. That is, the frequency of the first note in the scale is twice the frequency of the same note an octave below and half the frequency of the same note an octave above. Working with this octave repetition, the notes are treated as "pitch classes," meaning it does not matter what octave the note is in. Thus in our method, the pitches with the same note name from different octaves are wrapped (folded) into a single twelve-note vector, called a chroma vector.

Figure 2 shows octave equivalence on a piano keyboard:

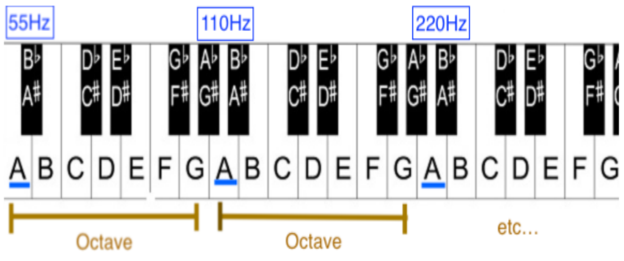


Figure 2: Octave Folds of Notes

Another aspect of chord recognition is handling the harmonics of each individual note. A musical note played by a real world instrument has harmonics related to the fundamental note (unless it is a generated sin or cosine tone). These harmonics do not fall under the same pitch class of the same note. This comes from the harmonic series, a concept from physics dealing with waves—in this case sound waves.

The fundamental frequency of a note, for example A = 440Hz, is the basis of the harmonics of that frequency, which are all integer multiples of the fundamental. The first overtone is found at 2 x 440Hz = 880Hz, which is also an A but an octave above. The second is 3 x 440Hz = 1320Hz; which is not an A, but an E, yet it is present when the note A (440Hz) is played on a real world instrument.

Furthermore, the sounds of different instruments are characterized by differing amplitude amounts for each harmonic, so saxophones will contribute different overtone amplitude strengths than flutes will.

This leads to difficulty when trying to identify the notes of a chord due to the harmonics of the "true" note should all be attributed to the note at the fundamental frequency. Yet, the harmonics give their energy to notes which are different from the fundamental frequency, lowering recognition accuracy. Methods to address this issue are discussed later in this paper.

Another issue that arises in chord classification is the subjective annotation of chords, since there can be multiple correct musical interpretations of a single chord, by traditional music theory rules. This means that musicians and experts will sometimes differ in their opinions about how to label a chord. This issue is especially present in Jazz music, due to the genre's prolific use of complex chords with more than four notes, and often leaving notes out of a chord being played. This is why chord classification can be an ambiguous target, sometimes resulting in discrepancy between separate human analyses [11].

We hope our project will be able to be evaluated using clear objective measurements such as classification accuracy, but we will likely run into subjective chords and chord progressions that are an issue for our algorithms and the humans determining the result [1]. The accuracy can be calculated by counting the number of matching samples between the predicted and the ground truth chords and dividing it by the total number of samples as done in [17].

It was our goal to develop an algorithm that reflects advancements made in the field of ACR, and provides an agreeable analysis of chords as determined by the musically perceptive members of our group.

4. METHODS

The modules that compose our system are connected as shown in Figure 3:

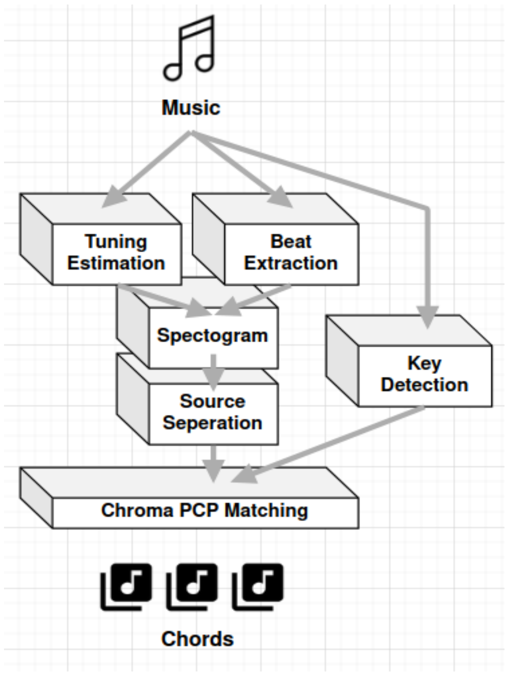


Figure 3: Module Connection Overview

This modular approach is constructed as a series of Python Notebooks. Many sections were developed independently from each other and modified later to operate together as a combined module (the final notebook is titled CombinedModule).

This process of development was somewhat accidental or forced by a lack of collaboration in Python Notebooks. Some services exist, such as Google Colaboratory, but we had issues with dependencies and disk space to store music files. Improvements for future work might include researching recommended forms of notebook collaboration, ideally methods that do not require a central server, such as Git.

Each module will be introduced and explored, then the combined system will be discussed.

Tuning Estimation is performed using Fast Fourier Transforms (FFT) across the entire song. A window size of 1 second was chosen, to reduce the number of calculations and to increase the frequency resolution. With each frequency magnitude output, we used peak picking to identify likely note fundamentals in the audio. These fundamentals were then compared to a standard tuning of 440 Hz, returning a tuning offset value. By taking the average (mode) tuning value across the entire song we found we could estimate the tuning of the entire song with relative accuracy. This tuning offset is used to calibrate our CQT for note extraction, further down the pipeline.

Source Separation is part of processing of audio for harmonic filtering and analysis by removing noise. Our system uses Harmonic-Percussive Source Separation (HPSS) to increase the harmonic resolution of the audio track. We used Librosa's built-in HPSS algorithm for this task.

Figure 4 shows the weighted harmonic mask applied to the spectrogram of a signal to filter out the noisy percussive portions of the signal. A similar mask for percussive sounds can be generated but was not used by our algorithm. This weighted mask is a matrix of magnitudes between 1 and 0 that, when applied to a spectrogram of the signal, removes frequency content which has been deemed to be percussive and not tonally relevant.

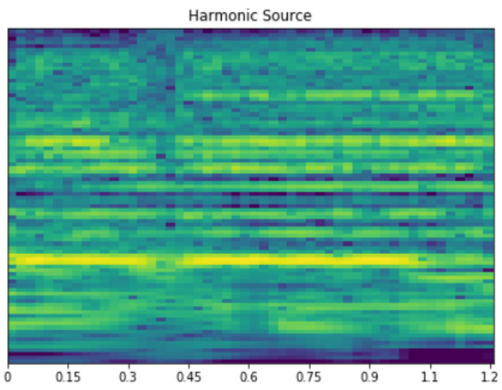


Figure 4: Spectrogram of a harmonic mask

Figure 5 shows an example of MIDI note magnitudes returned from librosa before and after the HPSS noise removal step. The blue plot is the original, and the orange plot shows how the noise floor is reduced and harmonic pitch resolution is increased. These pitch magnitudes are used in the Chroma Pitch Class Profile matching module to determine the chords.

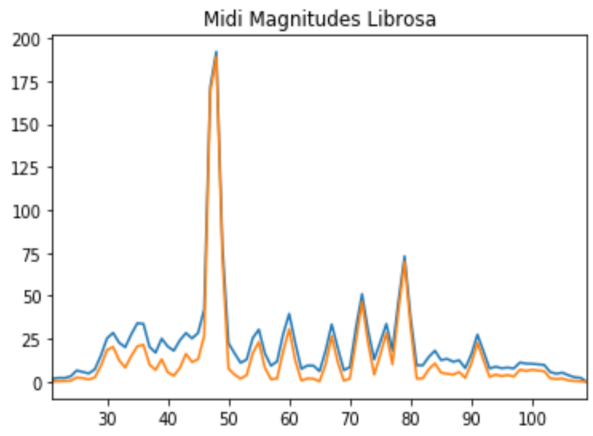
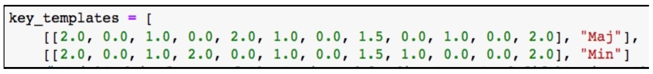


Figure 5: Blue/Orange shows before/after noise removal

Key Detection is performed by frequency analysis of the first and last 30 seconds of the song, using a CQT. This spectrum is folded into a single chroma vector, which is then compared to each maj and min key template, across all keys. The highest scoring key is deemed to be the key of the song, and the rest are ignored. The key templates are shown below in figure 6:



```
key_templates = [
    [[2.0, 0.0, 1.0, 0.0, 2.0, 1.0, 0.0, 1.5, 0.0, 1.0, 0.0, 2.0], "Maj"],
    [[2.0, 0.0, 1.0, 2.0, 0.0, 1.0, 0.0, 1.5, 1.0, 0.0, 0.0, 2.0], "Min"]
```

Figure 6: Key Detection Pattern Matching

Beat Tracking is used by our system to determine relative chord onsets. This process splits the music into the temporal components of the chord progression, so that the chords can be delivered one at a time to the Chord Identification module for analysis. There are two methods for this; onset-detection, based purely on the perceived "attack" of new notes/chords, and tempo-detection, which relies on predicting a repeated pattern of onsets (the "beat" of the music).

In most music, chords change on the beat. As long as the tempo is steady, the tempo-detection method will produce reliable results. In the event of dramatic phrasings and other tempo variation, however, this can lead to less accurate outcomes, as discussed in Section 5.

Shown in Figure 7 are a collection of graphs demonstrating the output of detected beats using the tempo-invariant onset-detection module (tested on music without a steady beat):

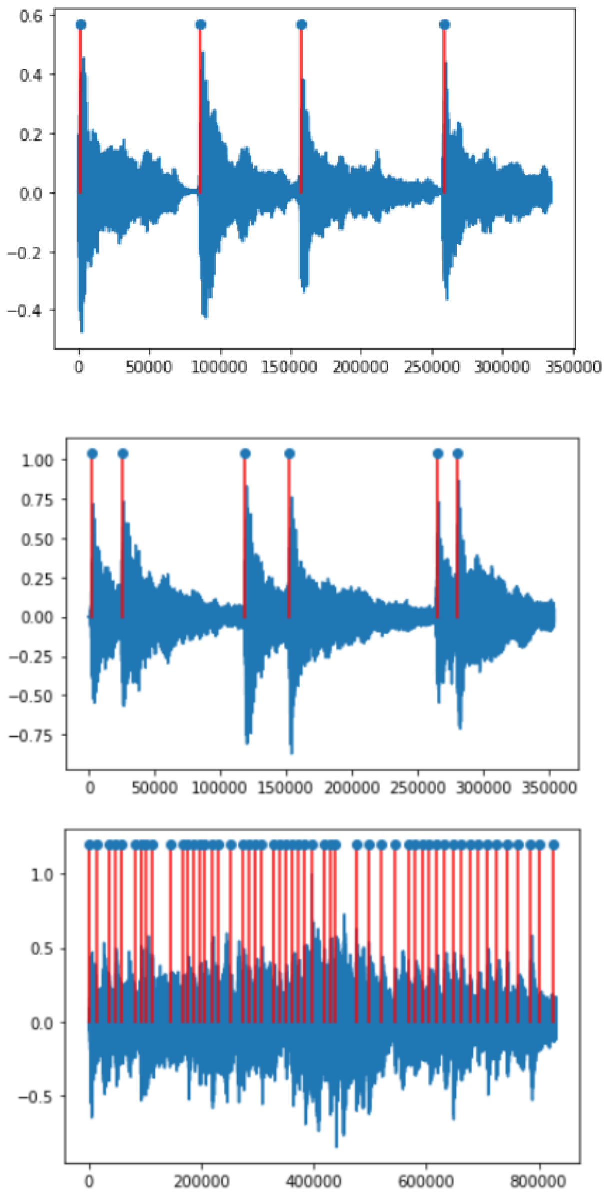


Figure 7: Beat Detection

Chroma Extraction is performed using a constant-Q transform (CQT) and pitch class folding, with the important step of extracting the bass note. We used Librosa's built in CQT, and provided the tuning offset from our Tuning Estimation module. Our implementation of the CQT looks at frequencies between 27.5Hz and 4434.92Hz, which correspond to the bottom and top notes of a standard piano keyboard, respectively.

At this stage, before folding the frequency information into a 12 note chroma, a bass note extraction is performed. This function analyzes peaks in the lower registers of the spectrum, midi notes 0 to 48, and determines the most likely bass notes for this excerpt of audio. This likelihood score is calculated from three factors: absolute magnitude of the note, noise floor of the surrounding frequency bins, and presence of 2nd and 3rd harmonics (midi note +12 and +19). A sample output is shown in Figure 7.5.

```
A (33)
total score:  75.44
abs_val:  76.08
noise:  0.4
harmonics:  0.39


A (45)
total score:  69.72
abs_val:  80.79
noise:  0.32
harmonics:  0.05


F (29)
total score:  19.34
abs_val:  33.27
noise:  1.04
harmonics:  0.2
```

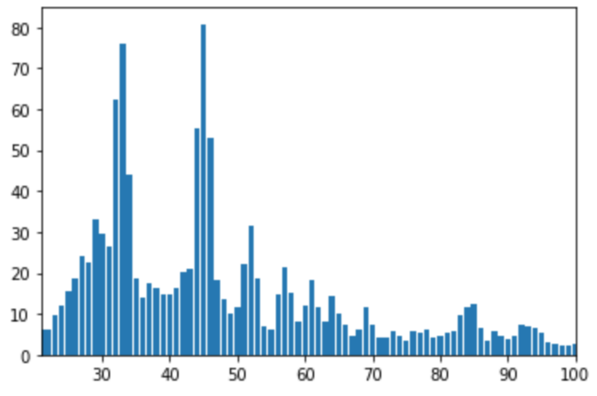


Figure 7.5: Bass note extraction

The function outputs the 3 highest scoring bass notes in the spectrum, which are later used as a starting point for chord identification.

Once the bass information has been extracted, the spectrum is folded into a 12 note chroma. This chroma represents the relative energy present for each pitch class, and therefore the harmonic content of the audio except.

Chord Identification is the last stage of our pipeline and uses Chroma PCP matching, calculating cosine similarity between chroma vectors for known chords, and the chroma vectors output by pitch-detection and pitch-folding.

This module was designed first in the system and was initially only capable of single-instrument chord recognition from a piano dataset [15] of unambiguous chords that include no background noise. Due to difficulties obtaining audio to match data sets, we relied on mostly manual annotation and evaluation to collect our results. This is further explained in the results section of our report.

As explained in Section 3, chords are formed from intervals of notes and the notes themselves "wrap around" every twelfth note of the chromatic scale (one octave). As such, it is easy to "rotate" an interval template around all twelve pitch classes to test for that quality of chord (major, minor, etc) on all twelve starting tonic notes, as shown in Figure 8 below:

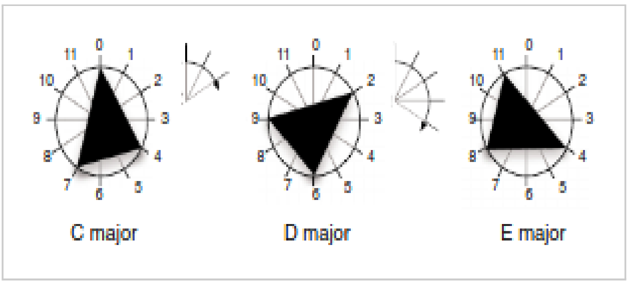
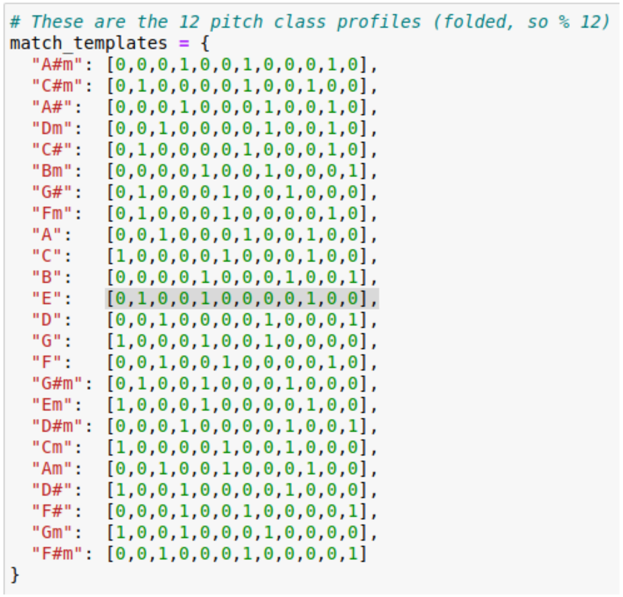


Figure 8: Rotating a Major Triad template

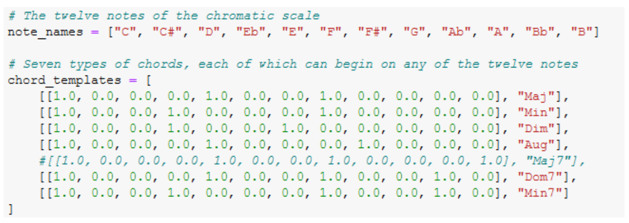
The following Figures 9 and 10 are drawn from our CQT pattern matching section of the Chord Identification module. Figure 9 shows the expanded set of chords being tested. Note that this only displays 24 chords; 12 Minor and 12 Major chords. Recall that in the Background section of this report chord vocabularies are introduced as a hierarchical system with major and minor chords being root nodes.



```
# These are the 12 pitch class profiles (folded, so % 12)
match_templates = {
  "A#m": [0,0,0,1,0,0,1,0,0,0,1,0],
  "C#m": [0,1,0,0,0,0,1,0,0,1,0,0],
  "A#":  [0,0,0,1,0,0,0,1,0,0,1,0],
  "Dm":  [0,0,1,0,0,0,0,1,0,0,1,0],
  "C#":  [0,1,0,0,0,0,1,0,0,0,1,0],
  "Bm":  [0,0,0,0,1,0,0,1,0,0,0,1],
  "G#":  [0,1,0,0,0,1,0,0,1,0,0,0],
  "Fm":  [0,1,0,0,0,1,0,0,0,0,1,0],
  "A":   [0,0,1,0,0,0,1,0,0,1,0,0],
  "C":   [1,0,0,0,0,1,0,0,0,1,0,0],
  "B":   [0,0,0,0,1,0,0,0,1,0,0,1],
  "E":   [0,1,0,0,1,0,0,0,0,1,0,0],
  "D":   [0,0,1,0,0,0,0,1,0,0,0,1],
  "G":   [1,0,0,0,1,0,0,1,0,0,0,0],
  "F":   [0,0,1,0,0,1,0,0,0,0,1,0],
  "G#m": [0,1,0,0,1,0,0,0,1,0,0,0],
  "Em":  [1,0,0,0,1,0,0,0,0,1,0,0],
  "D#m": [0,0,0,1,0,0,0,0,1,0,0,1],
  "Cm":  [1,0,0,0,0,1,0,0,1,0,0,0],
  "Am":  [0,0,1,0,0,1,0,0,0,1,0,0],
  "D#":  [1,0,0,1,0,0,0,0,1,0,0,0],
  "F#":  [0,0,0,1,0,0,1,0,0,0,0,1],
  "Gm":  [1,0,0,1,0,0,0,1,0,0,0,0],
  "F#m": [0,0,1,0,0,0,1,0,0,0,0,1]
}
```

Figure 9: Known Major and Minor Chords

Figure 10, by contrast, shows intervallic templates that can be rotated along the list of pitch classes, as described above. This section of code allows for the detection of 84 separate chords: 12 each of Major, Minor, Diminished, Augmented, Major 7th (currently commented out as Major 7th chords are uncommon), Dominant 7th, and Minor 7th.



```
# The twelve notes of the chromatic scale
note_names = ["C", "C#", "D", "Eb", "E", "F", "F#", "G", "Ab", "A", "Bb", "B"]

# Seven types of chords, each of which can begin on any of the twelve notes
chord_templates = [
    [[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0], "Maj"],
    [[1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0], "Min"],
    [[1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0], "Dim"],
    [[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0], "Aug"],
    #[[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0], "Maj7"],
    [[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0], "Dom7"],
    [[1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0], "Min7"]
]
```

Figure 10: Intervallic Chord Templates

In Figure 11 below, an E Major Chord is played from the piano dataset and is matched as such because the peaks in the pitch-class-profile best match the ones in the above tables. Note that the stacked lines are the harmonics of the notes at different frequencies.

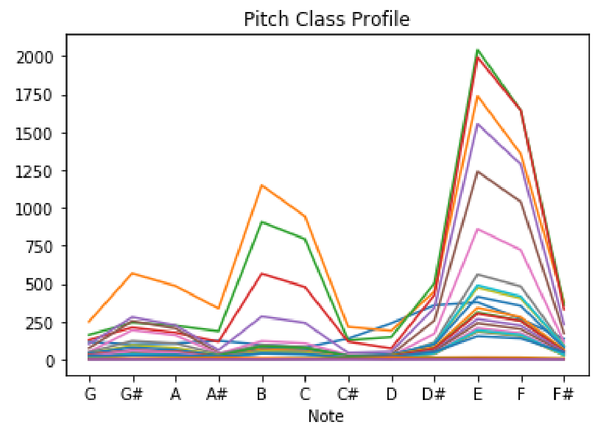
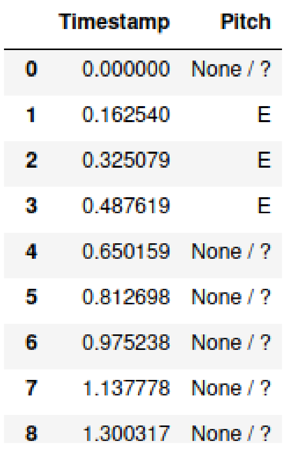


Figure 11:

Some timestamps of the above match are displayed in the following Figure 12:



	Timestamp	Pitch
0	0.000000	None / ?
1	0.162540	E
2	0.325079	E
3	0.487619	E
4	0.650159	None / ?
5	0.812698	None / ?
6	0.975238	None / ?
7	1.137778	None / ?
8	1.300317	None / ?

Figure 12: Example of PCP Matches vs Time

The **Combined System** collects all of the above modules into a single system (a single Python Notebook) totalling 644 lines of code.

This uses the extracted bass note as a basis for our PCP matching, to reduce the number of chord candidates being considered. Based on common practices and music theory, we assume that the bass note is always going to be a note from within the chord. For example, is the bass note is a C# the chord could be any chord that contains a C#. This narrows the scope of PCP matching and filters a large amount of noise from the results.

After receiving an array of likely chords from the PCP matching, the results are further sorted according to the detected key of the song. We first generate a set of chords that are likely to appear, based on the result from the key detection. Using this set of target chords, we prioritize chords from the array that are likely to appear in that key.

5. RESULTS

We used three datasets during development. These were the single chord piano dataset [15] used to test the original Chord Identification module on its own, as well as classical music from Bach [18] and an album of pop genre music from the Killers [19], which were used to test the final combined system.

5.1 POP MUSIC

To test chord recognition accuracy on modern pop music, we used certain songs from the album "Hot Fuss" by The Killers [19]. Due to our manual annotation system, we were not able to test large amounts of data and are left with incomplete results.

This analysis compared each output chord with a ground truth, counting exact matches as correct and anything else as incorrect. The total recall score across all songs tested was 58.3%.

Visually, it was clear from our outputs that smoothing is needed to improve results. Adjacent chords in an output file would often share roots or bass notes, but differ slightly in quality. These chords are often one single chord in the ground truth, that involves embellishment or small variations in arrangement. Smoothing would help mitigate this type of error

5.2 CLASSICAL BACH

For testing chord recognition of instruments in classical pieces, the following works from Bach were selected:

BWV {26,262,274,288} [18]. These pieces have sheet music published but no official chord labels. As such, it was necessary to formulate ground truth values by hand. This was a difficult and labour-intensive process, since the notation contains complex areas and had to be time stamped manually by ear to correspond to the recordings. Due to this difficulty, only 4 pieces were prepared for analysis:

- BWV 26 and BWV 262 on Piano
- BWV 274 and BWV 288 on Organ

These four pieces comprised a total of 169 chords to be found and identified.

The analysis recorded 4 types of results:

- Correctly-Identified Chords
- Incorrectly Identified Chords
- Missed Chords (chord listed in the ground truth was not isolated by the beat-separation module)
- Phantom Chords (chords that were separated out and analyzed despite not existing in the ground truth)

The test was first carried out using the tempo-based beat separation module, with the following results:

BWV:	***26***	***262***	***274***	***288***
Correct:	8	21	16	20
Incorrect:	32	23	12	21
Missed:	1	1	7	7
Phantom:	37	8	42	66

The test was then repeated, using the previous tempo-invariant onset detection method of beat-finding:

BWV:	***26***	***262***	***274***	***288***
Correct:	9	26	17	18
Incorrect:	32	19	14	25
Missed:	0	0	4	5
Phantom:	23	12	5	12

The total recall score was thus roughly 38.5% using the tempo-detection beat separation module (65/169), and 41.5% (70/169) for the previous onset-detection module.

All four Bach pieces featured expressive tempo variations that posed difficulties to the tempo-detection algorithm. In the first case, the rigid following of the pre-detected tempo resulted in a greater number of missed chords than in the second.

Additionally, Bach often employs out-of-chord passing tones on up-beats with the result that the detected "tempo" was often the eight note rather than the beat quarter, resulting in greatly increased instances of nonsense phantom chords. Even in the second case with the tempo-invariant beat divider, these up-beats were often strident enough to be picked out and analyzed on their own.

Some of the mistakes were understandable misreadings - A minor 7 (A, C, E, G) being misinterpreted as C major (C, E, G) for example. Others were possibly due to the recorded instrument being out of tune with itself (A major (A, C#, E) was instead detected as A# diminished (A#, C#, E -- if all three notes had been off by a half step, it would be presumed this was an error with our tuning detection module. With only one of them off, however, it suggests the piano in the recording might have had its A key tuned sharp).

And finally, it should be noted that a few of these mistakes were due to the very key-interpretation intended to aid in chord detection. Among other places, this can be observed in BWV 288 at around 32.25. Bach adds an "accidental" sharp, ending a phrase on a technically-out-of-key A major chord. Our module, which has deduced the piece is in the key of F (no C#), concludes that A minor chords are much more likely than A major ones and disbelieves its own "ears", blithely reporting an A minor chord.

6. DISCUSSION AND CONCLUSION

The pop results were a reasonable first attempt at a real world application of this system. It's clear that a number of the modules in our pipeline could be adjusted to benefit the pop results specifically.

For one, using tempo detection was a much more accurate means of approximating chord onsets for the pop genre. Given that the individual chord identification performed well alone, we can deduce that improving chord onset estimation even slightly would make significant improvements to the whole system output.

Another area for improvement, previously mentioned, would be chord smoothing. This would take adjacent chords with related qualities and smooth the progression of chord sequences according to probabilities. This is a very standard step in ACR systems and was part of our initial plan, but did not get implemented due to time constraints.

In the context of the complex Bach pieces, two issues come to the forefront. The first is the difficulty with accurately dividing the beats into chords for analysis. While dramatically-motivated tempo variations are harder for a computer to deal with than a steady tempo, this is an issue that can likely be dealt with by improving the beat-detection algorithm.

The second issue, that of passing tones played with weight equal to chord notes, requires further research as a straightforward solution does not immediately present itself. These notes can easily throw off the chord analysis if included, but separating them out is difficult without already knowing the chord to which they are being added.

Nonetheless, given the sheer number of chords available to choose from (given a fully accurate beat separation, a purely random classifier would have a recall of about 1.2%), roughly 40% correctness on such complex music is an encouraging number.

7. REFERENCES

[1] J. Pauwels; K. O'Hanlon; E. Gomez; M. B. Sandler, “20 Years of Automatic Chord Recognition from Audio” in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR, Delft, The Netherlands, pp. 54-63.*

[2] K. Lee; J. P. Bello, “Chord-Recognition: Automatic chord recognition with PCP (Pitch Class Profile)” on *Github*

[3] P. Rajpurkar; B. Girardeau, and T. Migimatsu, “A Supervised Approach To Musical Chord Recognition” in *Stanford University*

[4] A. R. Emmons, “Automatic Chord Recognition from Audio. *” Morris, Minnesota: University of Minnesota.*

[5] K. Ma, “Automatic Chord Recognition,” in *Department of Computer Science, University of Wisconsin-Madison*

[6] H. Cheng, Y. Yang, Y. Lin, I. Liao, and H. Chen, “Automatic Chord Recognition for Music Classification and Retrieval” in *ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October 2006, Proceedings.*

[7] K. Lee; M. Slaney, “Automatic Chord Recognition from Audio Using an HMM with Supervised Learning” in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR, Delft, The Netherlands, pp. 54-63.*

[8] V. Zenz, and A. Rauber, “Automatic Chord Detection Incorporating Beat and Key Detection” in *IEEE International Conference on Signal Processing and Communications, 2007. ICSPC 2007.*

[9] Quantcademy, “Hidden Markov Models - An Introduction,” *QuantStart*. [Online]. Available: https://www.quantstart.com/articles/hidden-markov-models-an-introduction/. [Accessed: 17-Oct-2020].

[10] Fujishima, T. “Realtime chord recognition of musical sound: A system using common lisp music.” in *Proceedings of the International Computer Music Conference, Beijing, China, 22–27 October 1999; pp. 464–467.*

[11] Hendrik Vincent Koops, W. Bas de Haas, John Ashley Burgoyne, Jeroen Bransen, Anna Kent-Muller, and Anja Volk. 2019. “Annotator Subjectivity in Harmony Annotations of Popular Music.” *Journal of New Music Research 48 (2019), 1–21.*

[12] J. C. Brown, “Musical fundamental frequency tracking using a pattern recognition method,” *J. Acous. Soc. Amer.*, vol. 92, no. 3, pp. 1394–1402, 1992.

[13] V. Emiya, R. Badeau and B. David, "Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle," *in IEEE Transactions on Audio, Speech, and Language Processing, vol. 18, no. 6, pp. 1643-1654, Aug. 2010, doi: 10.1109/TASL.2009.2038819.*

[14] Zhu, Y., Kankanhalli, M., & Gao, S. “Music Key Detection for Musical Audio.” 11th International Multimedia Modelling Conference (2005), 30-37.

[15] Peter Jones, "Piano Chords", Grand Piano – Fazioli Samples in Major & Minor. [Online] Available: https://ibeat.org/piano-chords-free/ [Accessed: 7-Nov-2020]

[16] Shan Anis, “Exploring Automatic Chord Recognition Algorithms,” *University of Rochester* [Online] Available: http://www2.ece.rochester.edu/~zduan/teaching/ece477/projects/2015/Shan_Anis_ReportFinal.pdf [Accessed: 12-Nov-2020]

[17] H. Lim; S. Rhyu; K. Lee, “Chord Generation from Symbolic Melody using BLSTM Networks” in *Music and Audio Research Group, Graduate School of Convergence Science and Technology, Seoul National University, Korea*

[18] Billam, Peter J, “Forty Bach Chorales arranged for keyboard”, *Creative Commons Attribution 4.0 International license*, Feb 17, 2014. Available: https://www.pjb.com.au/mus/arr/us/satb_chorales.pdf [Accessed November 26, 2020]

[19] The Killers, *Hot Fuss*, Cornerstone (Los Angeles, California), June 7, 2004.